

A. CAHIER DES CHARGES DU TP3- PROGRAMMATION EN ASSEMBLEUR DU MICROCONTROLEUR MC68HC11

I. Objectifs :

- ✓ Utiliser l'environnement du développement « MiniIDE » et le simulateur « Wookie » pour développer des programmes du microcontrôleur 68HC11.
- ✓ Utiliser et commander les sorties parallèles du 68HC11 (Port B) pour l'allumage séquentiel d'une ligne de LEDs (L0 à L7).
- ✓ Générer un signal carré, de rapport cyclique 50%, sur un bit d'un port du 68HC11 et signal carré de rapport cyclique 20% à l'aide du TIMER ;
- ✓ Ecrire un programme qui copie un octet entré sur le Port C et qui l'affiche sur le Port B.

II. Travaux réalisés :

1. Premier Programme :

Réalisation d'un programme qui commence par allumer la premier LED L0 (connecté au bit 0 du port B) et à la séquence suivante, la LED L1 (connecté au bit 1 du port B), et ainsi de suite (Sens d'allumage dirigé de la droite vers la gauche , séquence : L0,L1,...,L7) .Quand la dernier LED L7 est allumée on repart dans l'autre sens (sens d'allumage dirigé de la gauche vers la droite, séquence: (L7,L6 ,L5,L0), et on recommence indéfiniment.

1.1. *Ecriture et compilation du programme :*

On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'Programme LED.asm'.

Remarque :

- Si le programme s'écrit en couleurs, ça veut dire que l'écriture est correcte, par contre s'il est tout en noir, il existe une erreur quelque part.
- On peut ajouter des commentaires après chaque ligne de programme (;
commentaire)

```
MiniIDE - [Programme LED]
File Edit View Build Terminal Window Help

ORG $C000 ;Initialisation du PC par l'adresse $c000 (début du programme)

PORTB EQU $1004 ;Affecter à PORTB l'adresse $1004
LDAA #$01 ;Charger l'accumulateur A par la valeur $01

Up STAA PORTB ;Stocker le contenu de l'accumulateur A dans le PORTB ($1004)
JSR TEMPO ;Sauter à un sous programme TEMPO

LSLA ;Décaler les bits du portB vers la gauche

CMPA #$80 ;Comparer le contenu de l'accumulateur A avec $80
BNE Up ;Brancher à l'étiquette "UP" si le contenu de l'accumulateur A est différent de $80

Down STAA PORTB ;Stocker le contenu de l'accumulateur A dans le PORTB ($1004)
JSR TEMPO ;Sauter à un sous programme TEMPO
LSRA ;Décaler les bits du portB vers la droite

CMPA #$01 ;Comparer le contenu de l'accumulateur A avec $01
BNE Down ;Brancher à l'étiquette "DOWN" si le contenu de l'accumulateur A est différent de $01
BRA Up ;Branchement inconditionnel à "UP"

TEMPO LDAB #10 ;Charger l'accumulateur B par la valeur 10
ETIQ1 LDX $FFFF ;Charger l'index X par la valeur $FFFF
ETIQ2 DEX ;Décrémenter x de 1
BNE ETIQ2 ;Brancher à l'étiquette "ETIQ2" si le contenu de l'index X est différent de 0
DECB ;Décrémenter l'accumulateur B de 1
BNE ETIQ1 ;Brancher à l'étiquette "ETIQ1" si le contenu de l'accumulateur B est différent de 0
RTS ;Retourner du sous-programme au programme principal

Ln 1, Col 30
```

Image1 : Programme LED.asm

- ✓ On compile le programme avec 'Build Current'
- ✓ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✓ Programme LED.lst
- ✓ Programme LED. S19

```

Fichier  Edition  Format  Affichage  ?
C:\Users\said\Desktop\MiniIDE\Programme LED.lst -
generated by MGTEK Assembler ASM11 V1.22 Build 137 for WIN32 (x86) -
Sat Dec 17 15:21:43 2011

1:      =0000C000                                ORG      $C000
2:
3:      =00001004                                EQU      $1004
4:      C000 86 01                                [02]  PORTB EQU      LDAA    #$01
5:
6:      C002 B7 1004                                [04]  Up    STAA    PORTB
7:      C005 BD C01A                                [06]  JSR    TEMPO
8:
9:      C008 48                                    [02]  LSLA
10:
11:     C009 81 80                                    [02]  CMPA    #$80
12:     C00B 26 F5                                    [03]  BNE    Up
13:
14:     C00D B7 1004                                [04]  Down  STAA    PORTB
15:     C010 BD C01A                                [06]  JSR    TEMPO
16:     C013 44                                    [02]  LSRA
17:
18:     C014 81 01                                    [02]  CMPA    #$01
19:     C016 26 F5                                    [03]  BNE    Down
20:     C018 20 E8                                    [03]  BRA     Up
21:
22:     C01A C6 0A                                    [02]  TEMPO  LDAB    #10
23:     C01C CE FFFF                                [03]  ETIQ1  LDX     #$FFFF
24:     C01F 09                                    [03]  ETIQ2  DEX
25:     C020 26 FD                                    [03]  BNE    ETIQ2
26:     C022 5A                                    [02]  DECB
27:     C023 26 F7                                    [03]  BNE    ETIQ1
28:     C025 39                                    [05]  RTS

```

Image2 : Programme LED.lst

```

Fichier  Edition  Format  Affichage  ?
S0030000FC
S113C0008601B710048DC01A48818026F5B7100414
S113C0108DC01A44810126F520E8C60ACEFFFF09F7
S109C02026FD5A26F73943
S9030000FC

```

Image3 : Programme LED.S19

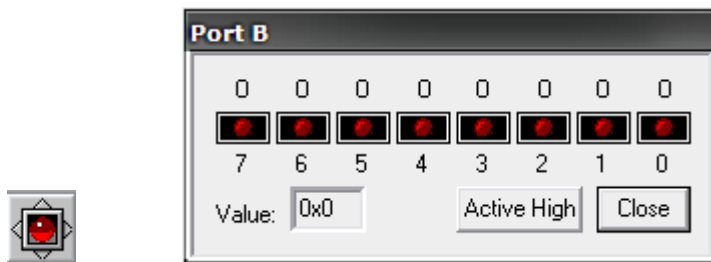
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

On lance le simulateur « Wookie »

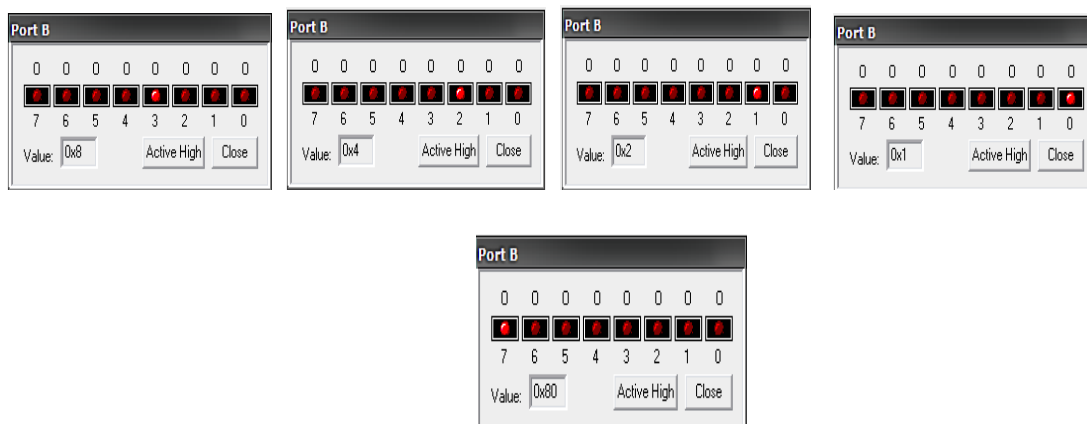
- ✓ On charge le fichier Programme LED.S19 avec 'Load .S19 file' dans le menu 'File'.
- ✓ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme (\$C000)
- ✓ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code]' et on tape OK.
- ✓ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :

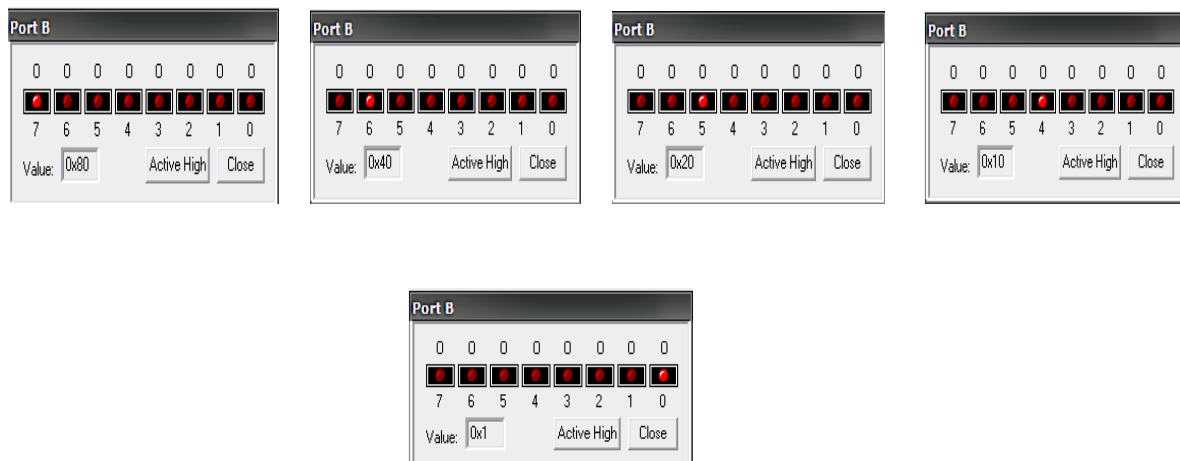


Après l'exécution

- Sens d'allumage dirigé de la droite vers la gauche, séquence : L0, L1, L2, L3..., L7



- Sens d'allumage dirigé de la gauche vers la droite, séquence : L7, L6, L5, L4..., L0



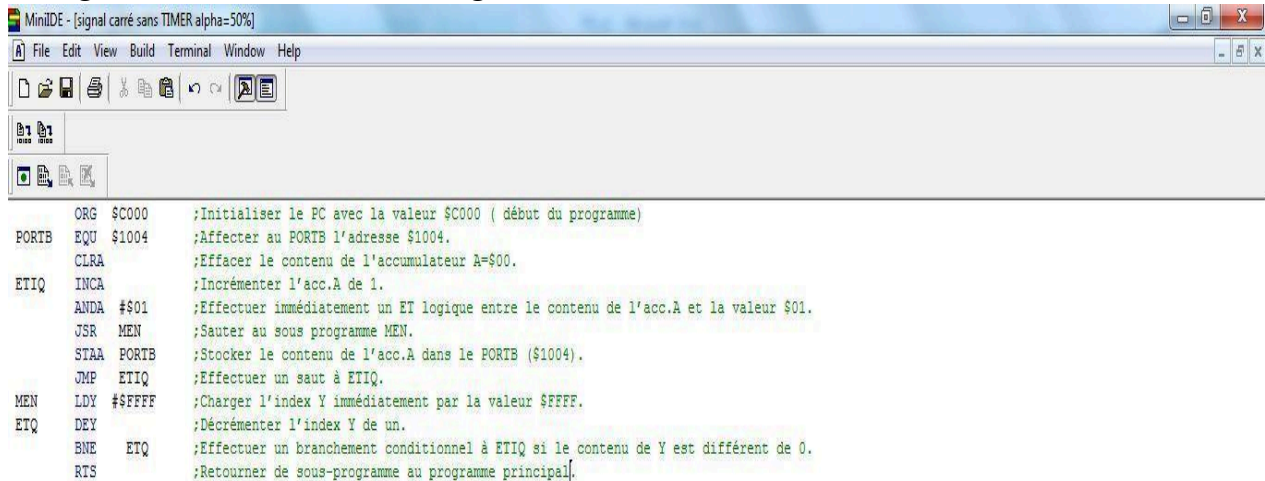
Le programme s'est bien effectué.

2. Deuxième Programme :

2 Premier sous programme: signal carré Sans utiliser le TIMER $\alpha=50$

a) Ecriture et compilation de programme

On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'signal carré Sans utiliser le TIMER $\alpha=50$.asm'.



```

MiniIDE - [signal carré sans TIMER alpha=50%]
File Edit View Build Terminal Window Help

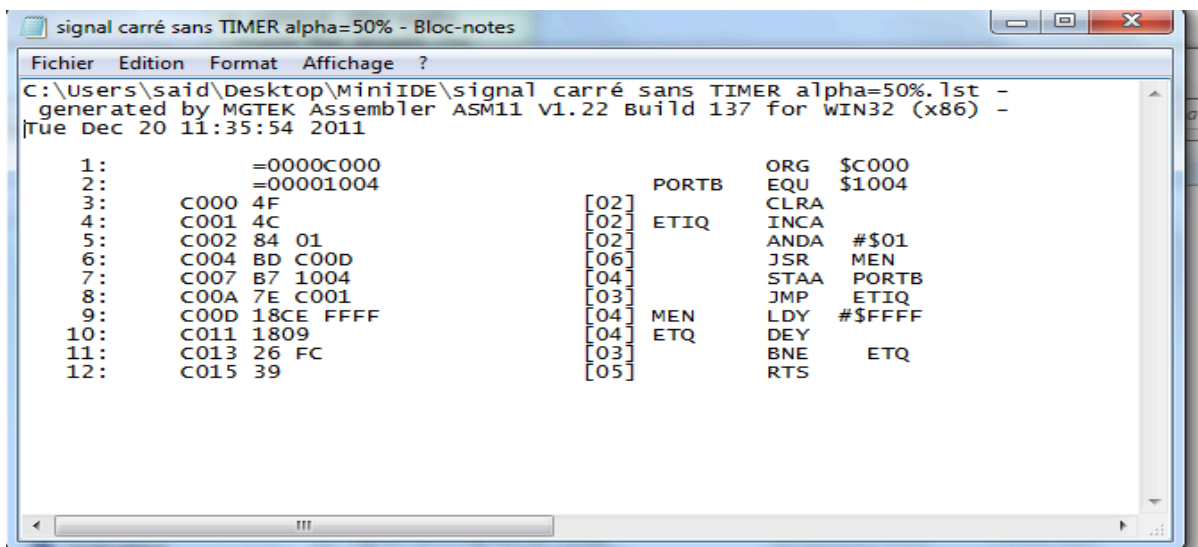
ORG $C000 ;Initialiser le PC avec la valeur $C000 ( début du programme)
PORTB EQU $1004 ;Affecter au PORTB l'adresse $1004.
CLRA ;Effacer le contenu de l'accumulateur A=$00.
ETIQ INCA ;Incrémenter l'acc.A de 1.
AND $01 ;Effectuer immédiatement un ET logique entre le contenu de l'acc.A et la valeur $01.
JSR MEN ;Sauter au sous programme MEN.
STAA PORTB ;Stocker le contenu de l'acc.A dans le PORTB ($1004).
JMP ETIQ ;Effectuer un saut à ETIQ.
MEN LDY $FFFF ;Charger l'index Y immédiatement par la valeur $FFFF.
ETQ DEY ;Décrémenter l'index Y de un.
BNE ETQ ;Effectuer un branchement conditionnel à ETIQ si le contenu de Y est différent de 0.
RTS ;Retourner de sous-programme au programme principal.
  
```

Image1 : Programme LED.asm

- ✓ On compile le programme avec 'Build Current'
- ✓ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✓ signal carré Sans utiliser le TIMER $\alpha=50$.lst
- ✓ signal carré Sans utiliser le TIMER $\alpha=50$. S19



```

signal carré sans TIMER alpha=50% - Bloc-notes
Fichier Edition Format Affichage ?
C:\Users\said\Desktop\MiniIDE\signal carré sans TIMER alpha=50%.lst -
generated by MGTEK Assembler ASM11 v1.22 Build 137 for WIN32 (x86) -
Tue Dec 20 11:35:54 2011

1:      =0000C000
2:      =00001004
3:      C000 4F
4:      C001 4C
5:      C002 84 01
6:      C004 BD C00D
7:      C007 B7 1004
8:      C00A 7E C001
9:      C00D 18CE FFFF
10:     C011 1809
11:     C013 26 FC
12:     C015 39

[02] PORTB ORG $C000
[02] EQU $1004
[02] CLRA
[02] INCA
[06] AND $01
[04] JSR MEN
[03] STAA PORTB
[04] JMP ETIQ
[04] MEN LDY $FFFF
[04] ETQ DEY
[03] BNE ETQ
[05] RTS
  
```

Image2 : signal carré Sans utiliser le TIMER $\alpha=50$.lst

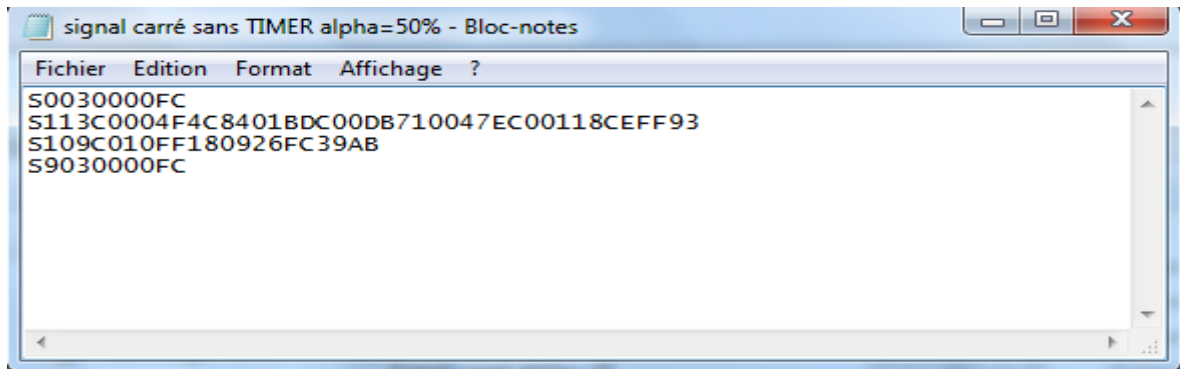


Image3 : signal carré Sans utiliser le TIMER $\alpha=50$.S19

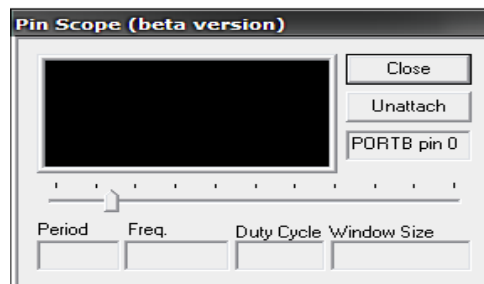
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

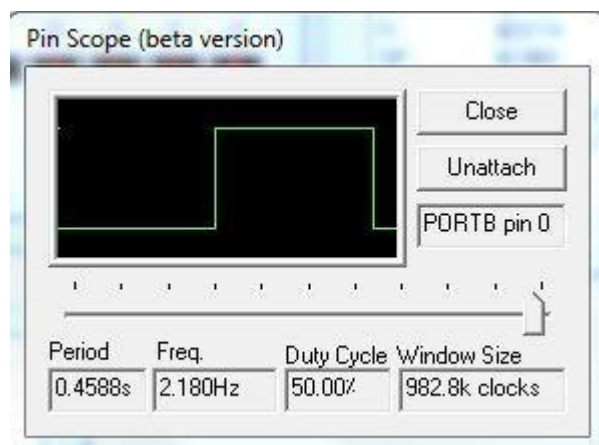
On lance le simulateur « Wookie »

- ✓ On charge le fichier *signal carré Sans utiliser le TIMER $\alpha=50$.S19* avec 'Load .S19 file' dans le menu 'File'.
- ✓ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme (\$C000)
- ✓ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code]' et on tape OK.
- ✓ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :



Après l'exécution :

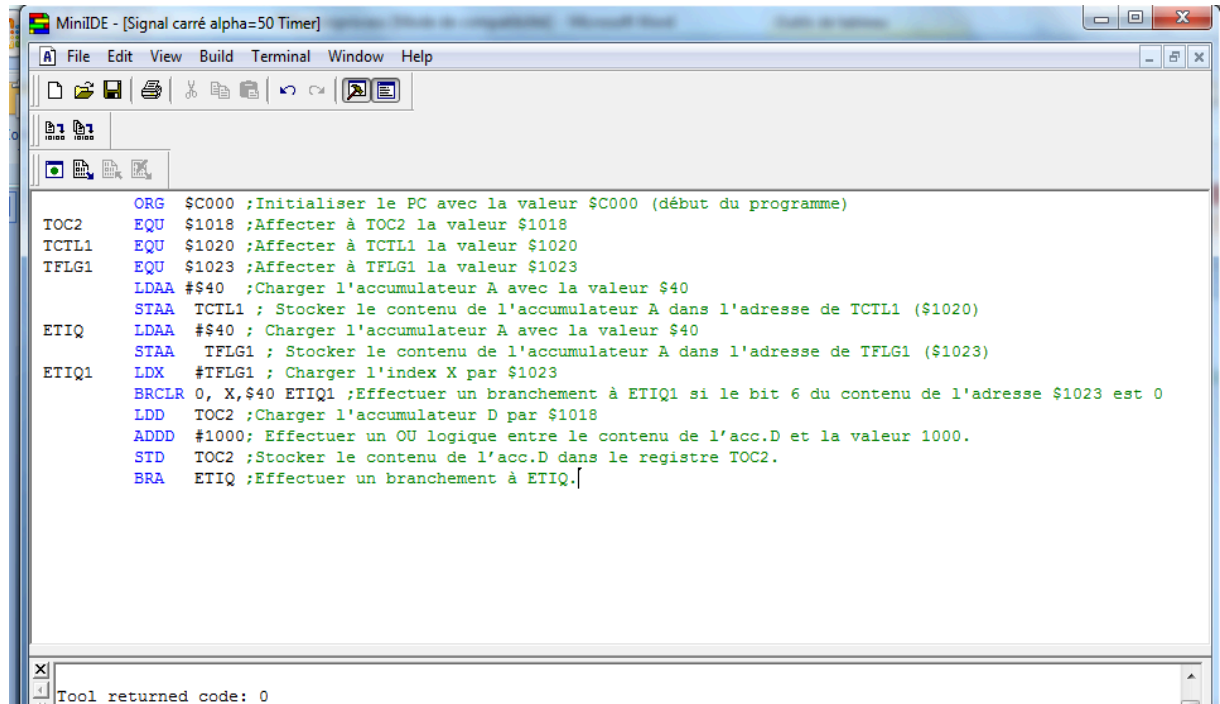


Après l'exécution on voit bien que notre programme est bien exécuté.
D'après la visualisation du signal on voit que sa fréquence est de 2.180Hz.

2 Deuxième sous programme: signal carré avec utilisation du TIMER $\alpha=50$

2.1. Ecriture et compilation du programme :

- ✓ On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'signal carré avec Timer alpha=50.asm'.



The screenshot shows the MiniIDE window titled "MiniIDE - [Signal carré alpha=50 Timer]". The menu bar includes File, Edit, View, Build, Terminal, Window, and Help. The toolbar contains icons for file operations and execution. The main text area displays the following assembly code:

```
ORG $C000 ;Initialiser le PC avec la valeur $C000 (début du programme)
TOC2 EQU $1018 ;Affecter à TOC2 la valeur $1018
TCTL1 EQU $1020 ;Affecter à TCTL1 la valeur $1020
TFLG1 EQU $1023 ;Affecter à TFLG1 la valeur $1023
LDAA #$40 ;Charger l'accumulateur A avec la valeur $40
STAA TCTL1 ; Stocker le contenu de l'accumulateur A dans l'adresse de TCTL1 ($1020)
ETIQ LDAA #$40 ; Charger l'accumulateur A avec la valeur $40
STAA TFLG1 ; Stocker le contenu de l'accumulateur A dans l'adresse de TFLG1 ($1023)
ETIQ1 LDX #TFLG1 ; Charger l'index X par $1023
BRCLR 0, X, $40 ETIQ1 ;Effectuer un branchement à ETIQ1 si le bit 6 du contenu de l'adresse $1023 est 0
LDD TOC2 ;Charger l'accumulateur D par $1018
ADDD #1000; Effectuer un OU logique entre le contenu de l'acc.D et la valeur 1000.
STD TOC2 ;Stocker le contenu de l'acc.D dans le registre TOC2.
BRA ETIQ ;Effectuer un branchement à ETIQ.
```

The status bar at the bottom indicates "Tool returned code: 0".

Image1 : signal carré avec TIMER $\alpha=50$.asm

- ✓ On compile le programme avec 'Build Current'
- ✓ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✓ signal carré avec TIMER $\alpha=50$.lst
- ✓ signal carré avec le TIMER $\alpha=50$. S19

```

C:\Users\said\Desktop\MiniIDE\Signal carré alpha=50 avec Timer.lst -
generated by MGTEK Assembler ASM11 v1.22 Build 137 for WIN32 (x86) -
Tue Dec 20 12:44:05 2011

1:      =0000C000                                ORG  $C000 |
2:      =00001018                                TOC2  EQU  $1018
3:      =00001020                                TCTL1 EQU  $1020
4:      =00001023                                TFLG1 EQU  $1023
5:      C000 86 40                                [02] LDAA #$40
6:      C002 B7 1020                                [04] STAA TCTL1
7:      C005 86 40                                [02] LDAA #$40
8:      C007 B7 1023                                [04] STAA TFLG1
9:      C00A CE 1023                                [03] LDX  #TFLG1
10:     C00D 1F 00 40 F9                            [07] BRCLR 0, X, $40 ETIQ1
11:     C011 FC 1018                                [05] LDD  TOC2
12:     C014 C3 03E8                                [04] ADDD  #1000
13:     C017 FD 1018                                [05] STD  TOC2
14:     C01A 20 E9                                [03] BRA  ETIQ

```

Image2 : signal carré avec TIMER $\alpha=50$.lst

```

S0030000FC

```

Image3 : signal carré avec TIMER $\alpha=50$.S19

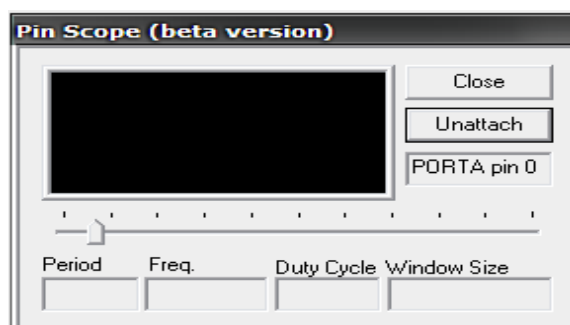
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

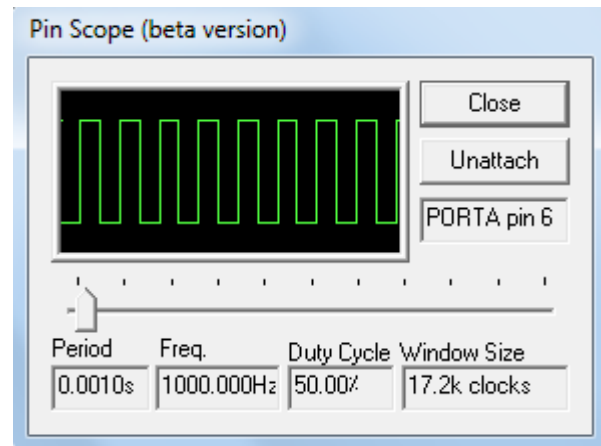
On lance le simulateur « Wookie »

- ✓ On charge le fichier *signal carré avec TIMER $\alpha=50$.S19* avec 'Load .S19 file' dans le menu 'File'.
- ✓ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme (\$C000)
- ✓ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code]' et on tape OK.
- ✓ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :



Après l'exécution :



Après l'exécution on voit bien que notre programme est bien exécuté.
D'après la visualisation du signal on voit que sa fréquence est de 1000Hz.

3 *Troisième sous programme: signal carré avec utilisation du TIMER $\alpha=20$*

2.2. *Ecriture et compilation du programme :*

- ✓ On saisit le programme dans l'environnement de développement « MinilDE », puis on l'enregistre sous l'extension .asm 'signal carré avec Timer alpha=20.asm'.

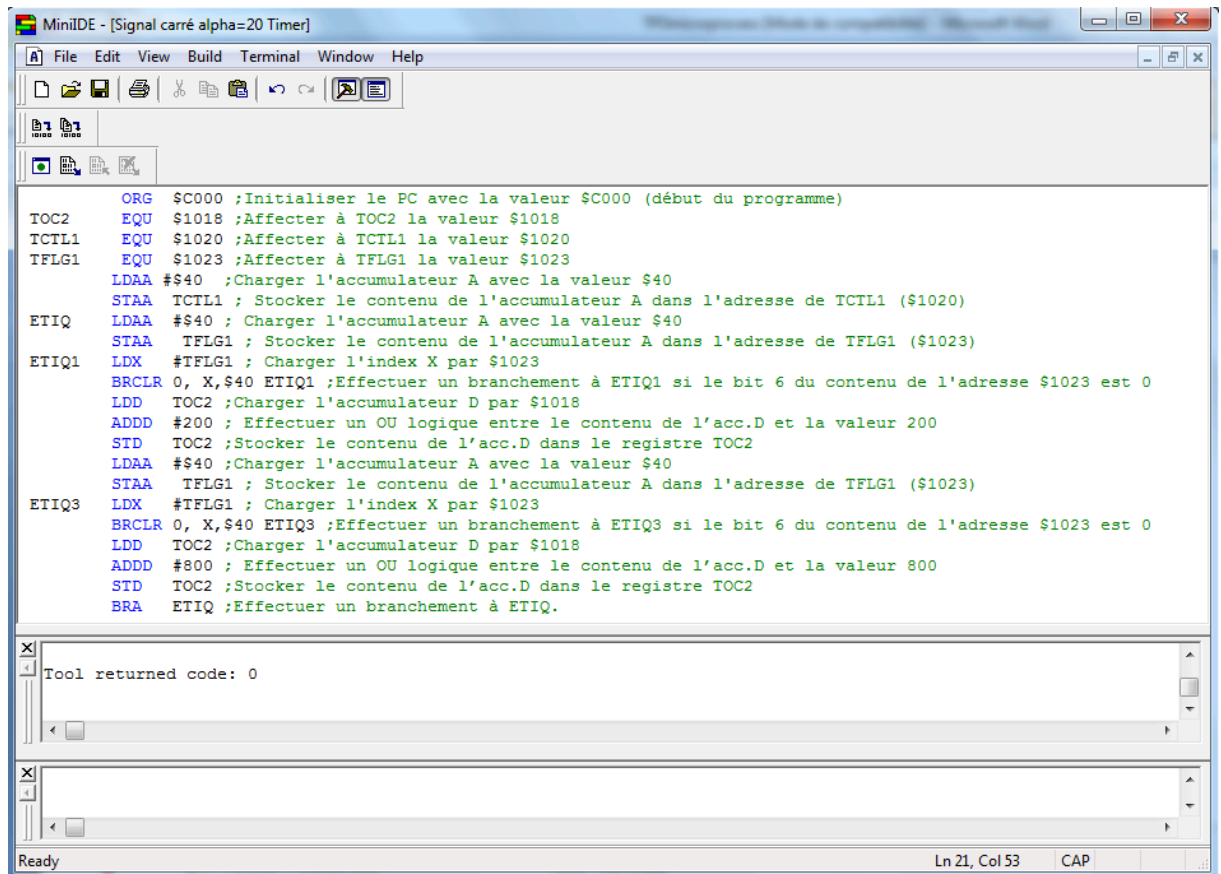


Image1 : signal carré avec *TIMER* $\alpha=20$.asm

- ✓ On compile le programme avec 'Build Current'
- ✓ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✓ *signal carré avec TIMER* $\alpha=20$.lst
- ✓ *signal carré avec le TIMER* $\alpha=20$. S19

```

Signal carré alpha=20 Timer - Bloc-notes
Fichier  Edition  Format  Affichage  ?
C:\Users\said\Desktop\MiniIDE\Signal carré alpha=20 Timer.lst -
generated by MGTEK Assembler ASM11 v1.22 Build 137 for WIN32 (x86) -
Wed Dec 21 12:19:43 2011

1:      =0000C000
2:      =00001018
3:      =00001020
4:      =00001023
5:      C000 86 40
6:      C002 B7 1020
7:      C005 86 40
8:      C007 B7 1023
9:      C00A CE 1023
10:     C00D 1F 00 40 F9
11:     C011 FC 1018
12:     C014 C3 00C8
13:     C017 FD 1018
14:     C01A 86 40
15:     C01C B7 1023
16:     C01F CE 1023
17:     C022 1F 00 40 F9
18:     C026 FC 1018
19:     C029 C3 0320
20:     C02C FD 1018
21:     C02F 20 D4

                                TOC2      EQU    $1018
                                TCTL1     EQU    $1020
                                TFLG1     EQU    $1023
                                [02]     LDAA  #$40
                                [04]     STAA  TCTL1
                                [02]     LDAA  #$40
                                [04]     STAA  TFLG1
                                [03]     ETIQ1  LDX   #TFLG1
                                [07]     BRCLR 0, X, $40 ETIQ1
                                [05]     LDD   TOC2
                                [04]     ADDD  #200
                                [05]     STD   TOC2
                                [02]     LDAA  #$40
                                [04]     STAA  TFLG1
                                [03]     ETIQ3  LDX   #TFLG1
                                [07]     BRCLR 0, X, $40 ETIQ3
                                [05]     LDD   TOC2
                                [04]     ADDD  #800
                                [05]     STD   TOC2
                                [03]     BRA   ETIQ

```

Image2 : signal carré avec TIMER $\alpha=20$.lst

```

Signal carré alpha=20 Timer - Bloc-notes
Fichier  Edition  Format  Affichage  ?
$0030000FC
$113C0008640B710208640B71023CE10231F00406F
$113C010F9FC1018C300C8FD10188640B71023CED1
$113C02010231F0040F9FC1018C30320FD10182032
$104C030D437
$9030000FC

```

Image3 : signal carré avec TIMER $\alpha=20$.S19

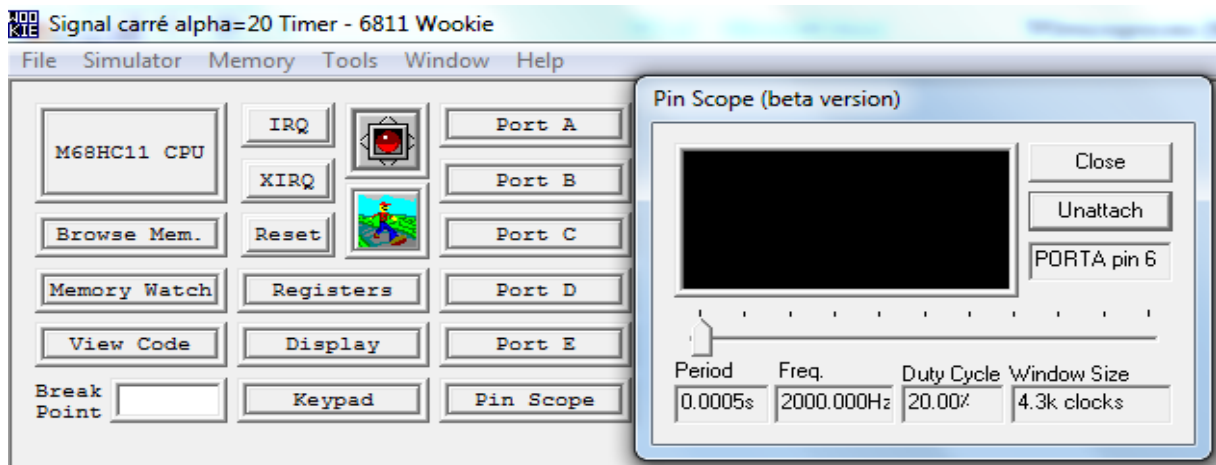
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

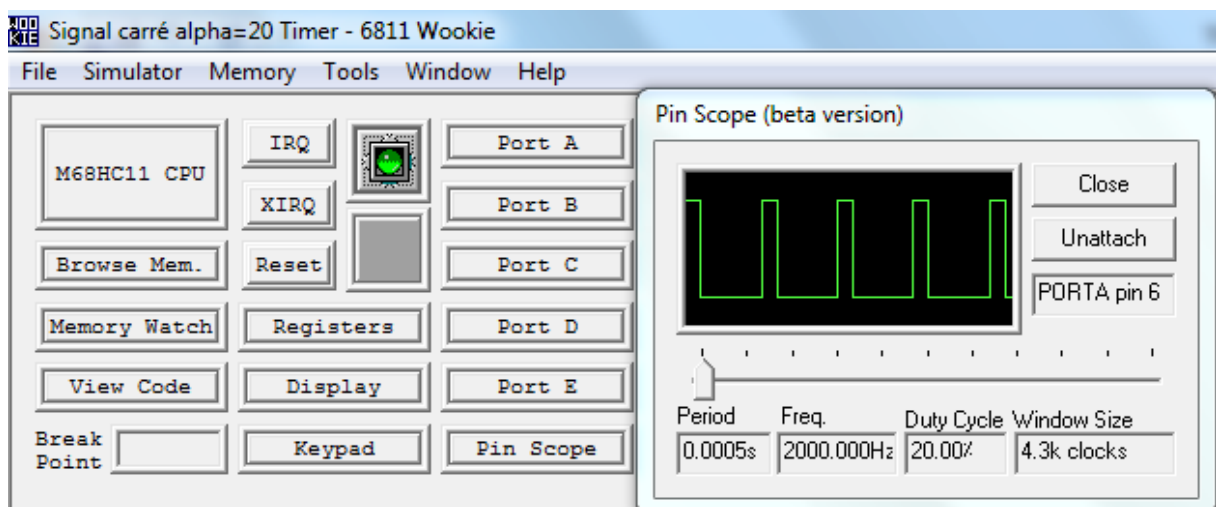
On lance le simulateur « Wookie »

- ✓ On charge le fichier *signal carré avec TIMER $\alpha=20$.S19* avec 'Load .S19 file' dans le menu 'File'.
- ✓ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme (\$C000)
- ✓ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code]' et on tape OK.
- ✓ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :



Après l'exécution :



Après l'exécution on voit bien que notre programme est bien exécuté.
D'après la visualisation du signal on voit que sa fréquence est de **2000Hz**.

3- Troisième programme :

3-1 – Ecriture et compilation du programme :

On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'Exercice3.asm'.

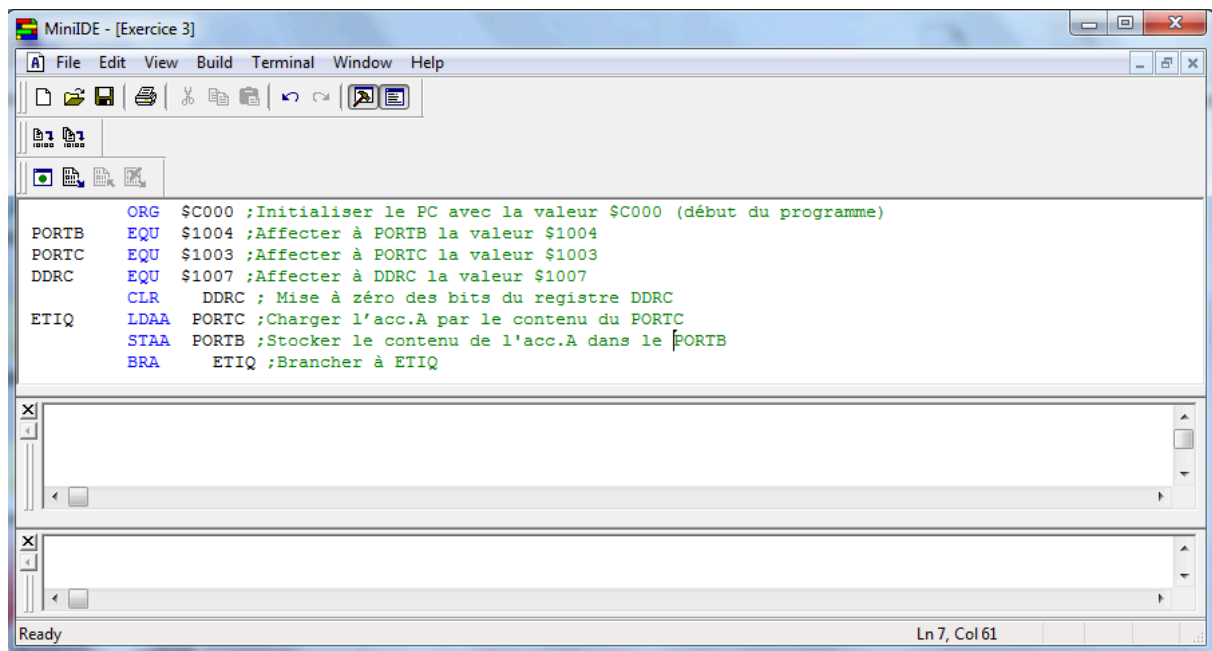


Image1 : Exercice3.asm

- ✓ On compile le programme avec 'Build Current'
- ✓ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✓ Exercice3.lst
- ✓ Exercice3. S19

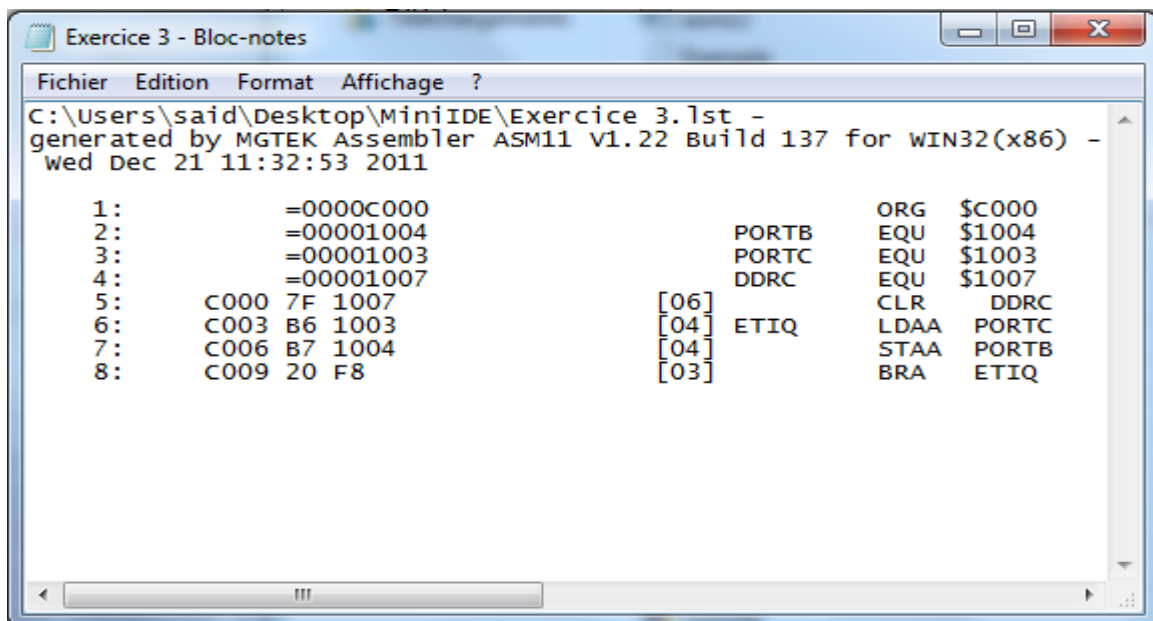


Image2 : Exercice3.lst

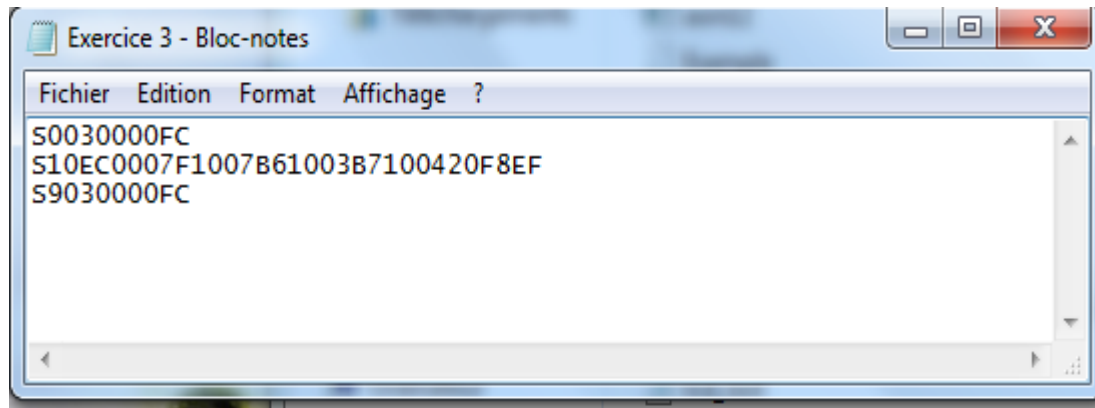


Image3 : Exercice3.S19

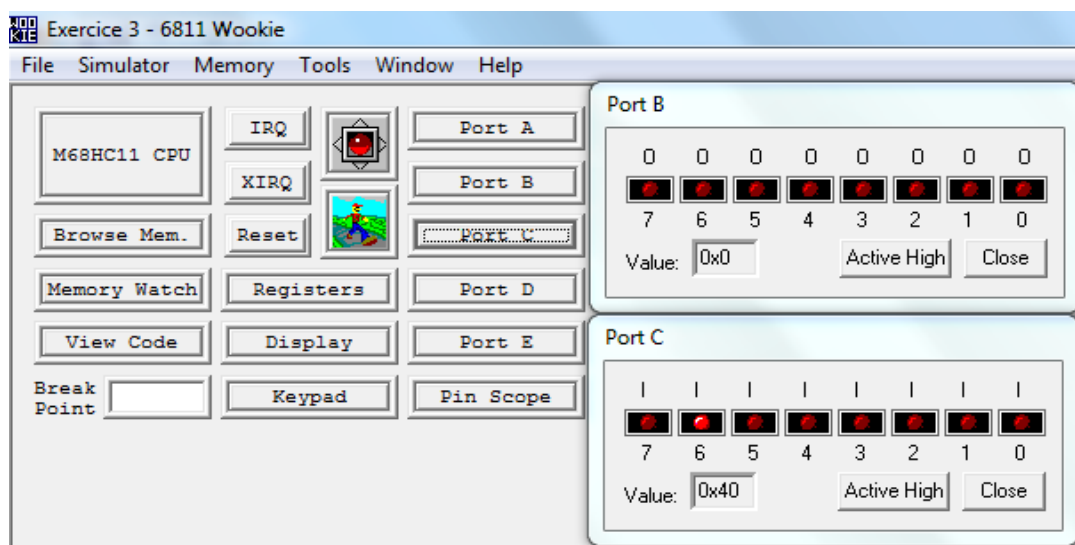
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

On lance le simulateur « Wookie »

- ✓ On charge le fichier *Exercice3.S19* avec 'Load .S19 file' dans le menu 'File'.
- ✓ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme (\$C000)
- ✓ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code]' et on tape OK.
- ✓ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

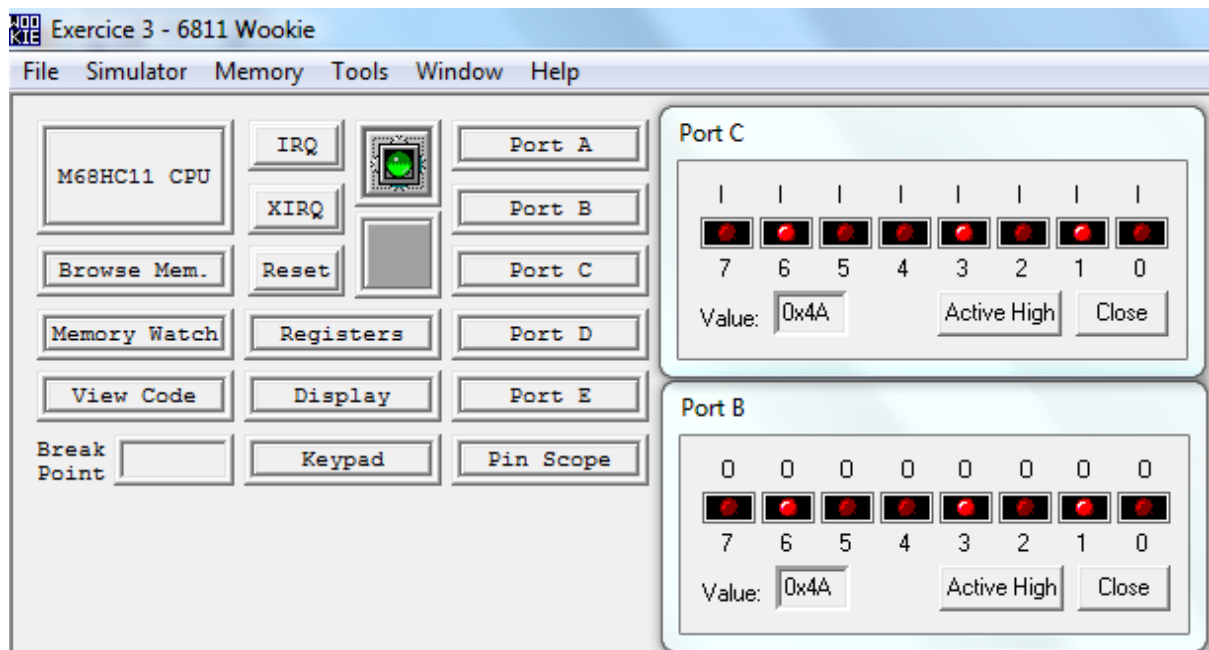
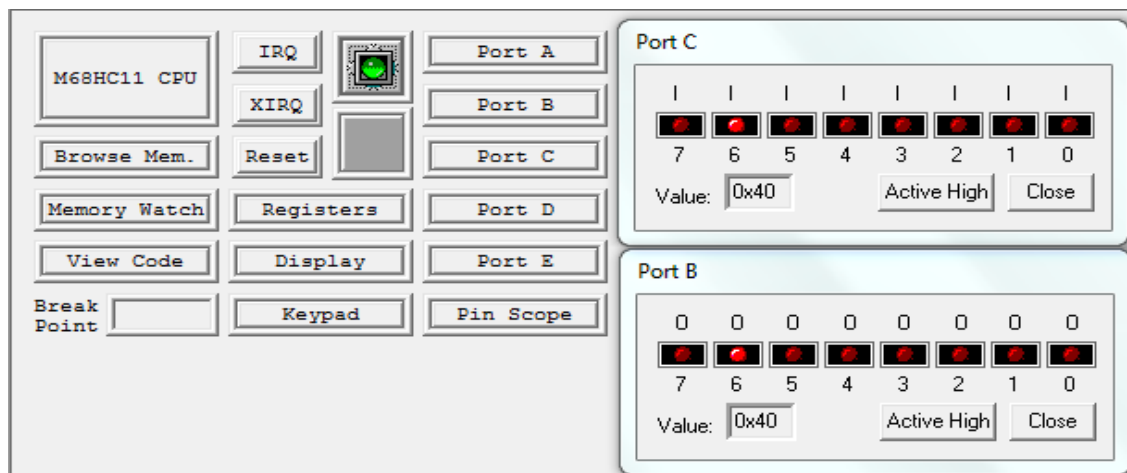
Avant l'exécution :



Remarque :

On voit bien que lorsqu'on a chargé le PORTC par une valeur le contenu du PORTB ne change pas, donc le programme n'est pas encore exécuté.

Après l'exécution :



Remarque :

A chaque fois qu'on charge le Port C par une valeur quelconque le Port B affiche la même valeur donc on peut dire que notre programme s'est bien exécuté.

CONCLUSION :

Le troisième TP consiste à développer des programmes en microcontrôleur MC68HC11 en utilisant son jeu des instructions, ses ports et les registres du TIMER comme des outils pour réaliser différents programme(on s'est servi de l'environnement du développement MiniIDE et le logiciel simulateur Wookie) :

- Pour allumage de LEDs (ou encore fonctionnement des LCD...)
- Génération de différents type de signaux avec ou sans utilisation du TIMER (on précise le rapport cyclique et la fréquence désirés ...)
- Stocker des valeurs différentes dans un registre unidirectionnel (comme le PORTB) en changeant les valeurs d'un registre bidirectionnel (comme le PORT C)
- ... etc.

